**Содержание**

[Введение 5](#_Toc166575554)

[1 Аналитический обзор литературы 6](#_Toc166575555)

[1.1 Аналог «Mebel.by» 6](#_Toc166575556)

[1.2 Аналог «Stolplit» 7](#_Toc166575557)

[1.3 Аналог «Lazurit» 8](#_Toc166575558)

[2 Проектирование приложения 9](#_Toc166575559)

[2.1 Проектирование базы данных 9](#_Toc166575560)

[2.2 Описание информационных объектов и ограничений целостности 10](#_Toc166575561)

[2.2.1 Коллекция users 10](#_Toc166575562)

[2.2.2 Коллекция products 10](#_Toc166575563)

[2.2.3 Коллекция favorites 10](#_Toc166575564)

[2.2.4 Коллекция orders 10](#_Toc166575565)

[2.3 Проектирование мобильного приложения 11](#_Toc166575566)

[3 Реализация приложения 13](#_Toc166575567)

[3.1 Технические средства разработки 13](#_Toc166575568)

[3.2 Разработка мобильного приложения 14](#_Toc166575569)

[4 Тестирование приложения 16](#_Toc166575570)

[5 Руководство по использованию 17](#_Toc166575571)

[5.1 Руководство гостя 17](#_Toc166575573)

[5.2 Руководство пользователя 22](#_Toc166575574)

[5.3 Руководство администратора 26](#_Toc166575575)

[Заключение 28](#_Toc166575576)

[Список использованных источников 29](#_Toc166575577)

[ПРИЛОЖЕНИЕ А 27](#_Toc166575578)

[ПРИЛОЖЕНИЕ Б 28](#_Toc166575579)

# Введение

Мобильные приложения играют ключевую роль в современном мире розничной торговли мебелью. Они обеспечивают удобство и доступность для покупателей, позволяя им взаимодействовать с брендом в любое время и в любом месте. Это особенно важно в эпоху цифровизации, когда большинство людей используют смартфоны для доступа к интернету. Мобильные приложения также могут улучшить взаимодействие с клиентами, предоставляя персонализированный опыт и улучшая обслуживание клиентов.

Для интернет-магазина мебели мобильное приложение может принести множество преимуществ. Во-первых, оно может увеличить продажи, предоставляя клиентам удобный способ просмотра и покупки мебели. Во-вторых, оно может помочь в укреплении отношений с клиентами, предлагая им персонализированные рекомендации и специальные предложения. Наконец, мобильное приложение может помочь в сборе ценной информации о предпочтениях и поведении покупателей, которую можно использовать для улучшения ассортимента и услуг. В целом, мобильное приложение может стать мощным инструментом для увеличения конкурентоспособности и роста бизнеса в сфере продажи мебели.

Были поставлены следующие цели разработки проекта:

* Реализовать функции гостя, пользователя и администратора;
* Реализовать регистрацию и авторизацию пользователей;
* Обеспечить управление товарами;
* Реализовать корзину пользователя;
* Имплементировать фильтрацию товаров;
* Обеспечить управление профилем для пользователей.

Основной целью курсового проекта является разработка приложения для магазина мебели. В качестве языка был выбран Dart в связке с фреймворком Flutter. В качестве провайдера авторизации используется Firebase, а для локальной бд был выбран Hive, для удаленной – Firebase Firestore в связке с Firebase Storage.

# 1 Аналитический обзор литературы

Обзор аналогов является критически важным этапом в процессе разработки программного обеспечения. Он позволяет разработчикам получить представление о существующих решениях, их функциональности, преимуществах и недостатках. Это помогает избежать повторения ошибок, сделанных другими, и определить уникальные особенности, которые можно внедрить в новое ПО. Аналоги рассмотрены ниже.

## **Аналог «Mebel.by»**

Mebel.by - это официальное приложение, Mebel.by - это официальное приложение для интернет-магазина мебели, разработанное компанией Onliner LLC. Оно предназначено для удобного выбора мебели и совершения покупок на мобильных телефонах и планшетах. В каталоге Mebel.by представлено множество категорий мебели, включая более 700 000 товаров, где покупатели могут найти подробные характеристики, фотографии, отзывы других покупателей и цены.

Каталог Mebel.by является маркетплейсом, что означает, что он содержит предложения от многих продавцов мебели, что предоставляет покупателям большую свободу выбора. Это позволяет клиентам сравнивать различные варианты и выбирать те, которые наилучшим образом соответствуют их потребностям и бюджету., что он содержит предложения от многих продавцов, что предоставляет большую свободу выбора покупателю.

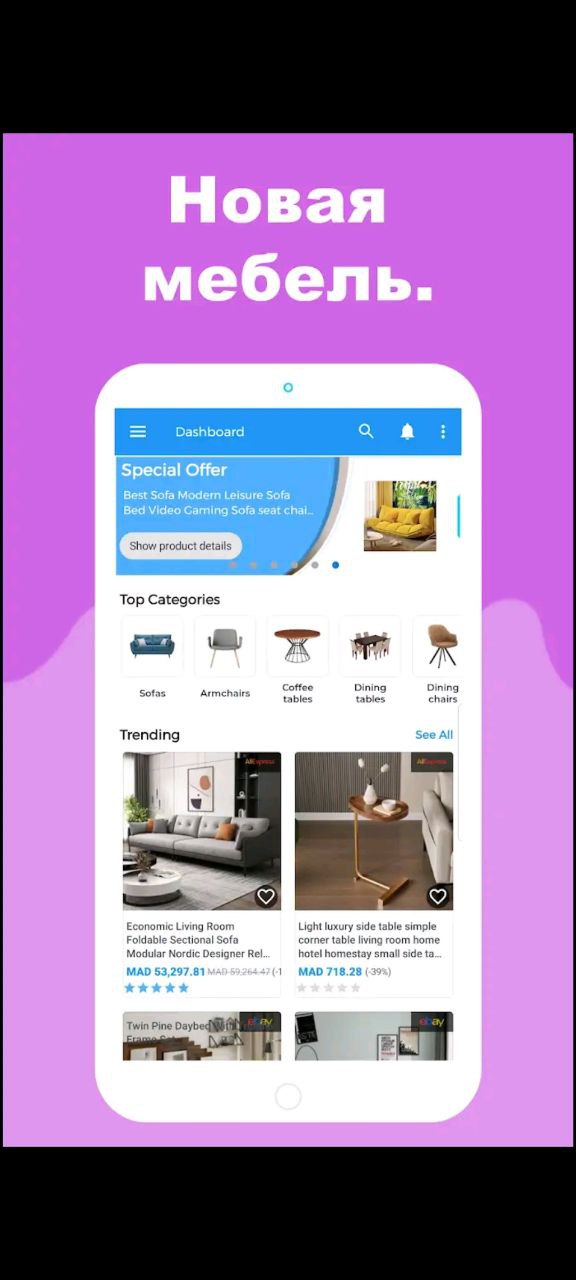


Рисунок 1.1 – Приложение «Mebel.by»

Каталог Mebel.by также обладает рядом преимуществ, которые делают его удобным инструментом для выбора и покупки мебели. Он предлагает широкий ассортимент мебели, что делает его одним из самых обширных каталогов в байнете. Приложение адаптировано под мобильные устройства, что делает его интерфейс простым и понятным. Кроме того, приложение позволяет сравнивать различные варианты мебели, что облегчает процесс выбора.

Однако, несмотря на все его преимущества, Каталог Mebel.by имеет и свои недостатки. Некоторые функции, доступные на веб-сайте, могут отсутствовать в мобильном приложении. Это означает, что пользователи могут не получить полный функционал, доступный на веб-сайте. Это может включать в себя некоторые опции сравнения, детальные характеристики товаров или определенные способы оплаты. Несмотря на это, Mebel.by продолжает улучшать свое мобильное приложение, чтобы обеспечить наилучший пользовательский опыт.

## **1.2 Аналог** «Stolplit»

Мобильное приложение Stolplit - приложение, представленное на рисунке 1.2, предназначенное для удобного выбора и покупки мебели на мобильных устройствах. Оно представляет собой полноценный каталог мебели с подробными характеристиками, фотографиями и отзывами. Дизайн приложения сделан похожим на сайт, что обеспечивает привычный и удобный интерфейс для пользователей. Это обеспечивает покупателям возможность легко просматривать и сравнивать различные варианты мебели, делая процесс выбора более простым и приятным.

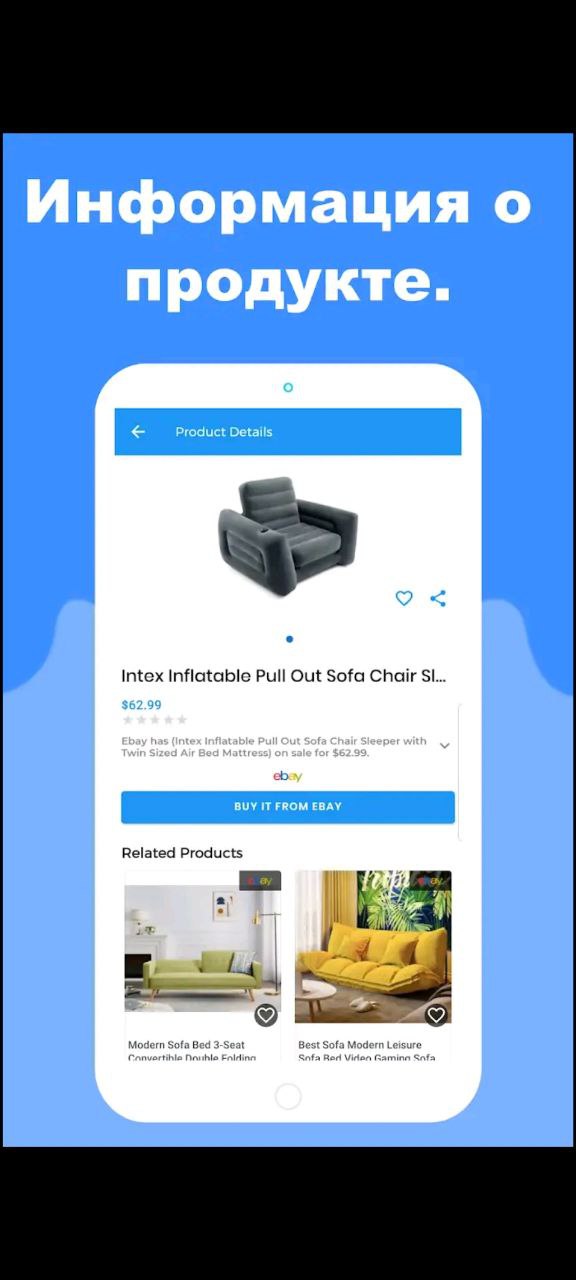


Рисунок 1.2 – Приложение «Stolplit»

Среди преимуществ приложения Stolplit можно выделить быстрый поиск мебели. Это позволяет пользователям получить всю необходимую информацию о мебели прямо в магазине. Кроме того, в приложении есть инструмент для сравнения различных вариантов мебели, что помогает в выборе. Пользователи могут отслеживать изменения цен и наличие товаров, добавляя их в список избранного.

Как и любое другое приложение, мобильное приложение Stolplit имеет свои недостатки. Несмотря на то, что приложение доступно для пользователей Android, релиз iOS-версии приложения мог произойти позже. Это могло быть неудобно для пользователей Apple. Однако, команда Stolplit постоянно работает над улучшением своего приложения, чтобы обеспечить наилучший пользовательский опыт для всех своих клиентов, независимо от их операционной системы.

1.3 Аналог «Lazurit»

Мобильное приложение Lazurit - это официальное приложение от компании Lazurit, представленное на рисунке 1.3, предназначенное для удобного выбора и покупки мебели на мобильных устройствах. Оно представляет собой каталог мебели с подробными характеристиками, фотографиями и отзывами. Это обеспечивает покупателям возможность легко просматривать и сравнивать различные варианты мебели, делая процесс выбора более простым и приятным.

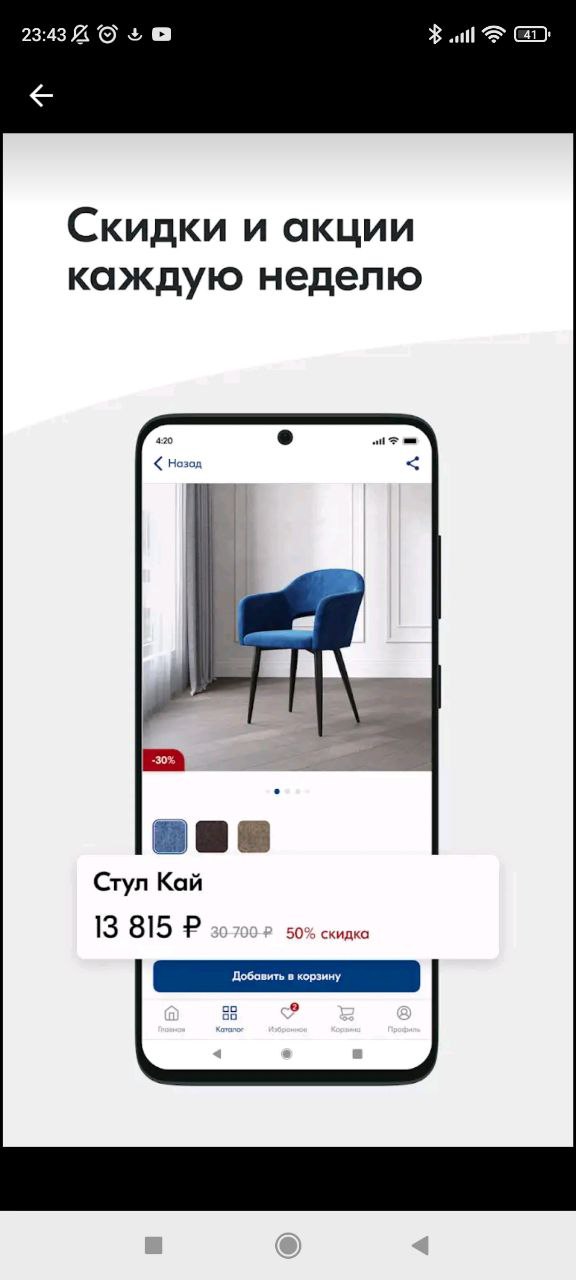


Рисунок 1.3 – Приложение «Lazurit»

# 2 Проектирование приложения

Проектирование программного обеспечения - это процесс создания плана для разработки программного продукта. Оно включает в себя определение функциональности, структуры, архитектуры и интерфейсов программного продукта. Проектирование ПО также учитывает такие факторы, как производительность, надежность, удобство использования и безопасность. Этот процесс является ключевым этапом в жизненном цикле разработки программного обеспечения и служит основой для написания кода.

## Проектирование базы данных

Проектирование модели данных для NoSQL базы данных - это процесс определения структуры данных, которая наиболее эффективно отвечает требованиям приложения. В отличие от традиционных реляционных баз данных, NoSQL базы данных обычно не используют структуру таблиц и не требуют жесткой схемы данных. Это позволяет создавать более гибкие и масштабируемые модели данных. При проектировании модели данных для NoSQL базы данных важно учитывать тип базы данных (документо-ориентированная, колоночная, ключ-значение или графовая), требования к производительности, масштабируемости и надежности, а также специфику приложения, которое будет использовать эту базу данных.

Для данного курсового проекта была спроектирована база данных, логическая модель базы данных представлена на рисунке 2.1.

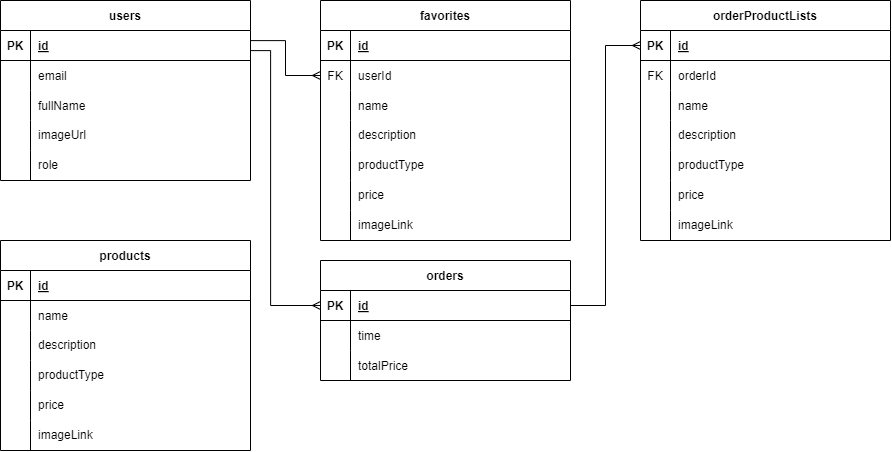


Рисунок 2.1 – Модель БД

Созданная база данных содержит в себе 4 коллекции: users, products, favorites, orders. Коллекция orders содержит внутри себя список, обозначенный на диаграмме как orderProductList.

2.2 Описание информационных объектов и ограничений целостности

Далее будет описана структура каждой коллекции. В коллекции users хранятся данные для идентификации пользователей, зарегистрированных в приложении, products – данные товарах, favorites для хранения избранных товаров и orders для хранения заказов.

### **2.2.1 Коллекция users**

Коллекция состоит из следующих полей:

* id. Хранит уникальный идентификатор пользователя;
* email. Хранит уникальный email пользователя;
* name. Хранит имя пользователя;
* imageUrl. Хранит ссылку на фотографию профиля пользователя;
* role. Хранит роль пользователя.

### **2.2.2 Коллекция products**

Коллекция products состоит из следующих полей:

* id. Хранит уникальный идентификатор товара;
* description. Хранит описание товара;
* productType. Хранит категорию товара;
* price. Хранит цену товара;
* imageLink. Хранит ссылку на фотографию товара.

### **2.2.3 Коллекция favorites**

Коллекция favorites является вложенной коллекцией users. В состав коллекции входят следующие поля:

* id. Хранит уникальный идентификатор товара;
* description. Хранит описание товара;
* productType. Хранит категорию товара;
* price. Хранит цену товара;
* imageLink. Хранит ссылку на фотографию товара.

### **2.2.4 Коллекция orders**

Коллекция orders является вложенной коллекцией users. В состав коллекции входят следующие поля:

* productList. Хранит список товаров заказа;
* totalPrice. Хранит общую сумму заказа;
* time. Хранит время оформления заказа.

## 2.3 Проектирование мобильного приложения

Проектирование мобильных приложений - это сложный процесс, который включает в себя несколько ключевых аспектов, включая проектирование страниц и навигацию.

Проектирование страниц в мобильных приложениях требует учета ограниченного пространства экрана и специфики взаимодействия пользователя с мобильным устройством. Каждая страница должна быть интуитивно понятной и содержать только самую важную информацию, чтобы не перегружать пользователя. Дизайн должен быть простым и чистым, с хорошим балансом между текстом, изображениями и пространством.

Навигация в мобильных приложениях также играет важную роль. Она должна быть простой и интуитивной, чтобы пользователи могли легко переходить от одной страницы к другой. Обычно используются стандартные элементы навигации, такие как меню, вкладки и кнопки “Назад”. Однако важно учитывать контекст приложения и потребности пользователей при проектировании системы навигации.

При первом входе в приложения пользователя встречает экран, описывающий как пользоваться приложением. После этого он попадает на главный экран.

На главном экране представлены текущие предложения с возможностью перейти к каталогу товаров. В нем пользователь может выбрать товар, чтобы перейти на его страницу. Также, если пользователь является администратором, то он может перейти на страницу добавления нового товара. Внизу отображается таб-бар с навигацией на страницу профиля и корзины.

На странице товара представлена информация о товаре, с возможностью добавить товар в корзину и избранное. Администратор может изменить или удалить товар.

На странице корзины отображаются товары корзины, и если пользовател авторизован, то он может перейти на страницу оформления заказа для того, чтобы подтвердить его.

На странице профиля отображается текущая информация о пользователе или гостевая информация. Если пользователь авторизован, то он может просмотреть историю заказов. Также есть возможность просмотреть текущее избранные товары. Для авторизованных пользователей отображается возможность выйти, а для неавторизованных – войти. На странице входа можно перейти на страницу регистрации. Авторизованные пользователи могут редактировать свое имя и менять фотографию профиля.

В профиле пользователь может просмотреть свои персональные данные, а также редактировать профиль и выйти из аккаунта.

На рисунке 2.2 представлена схема навигации гостя.

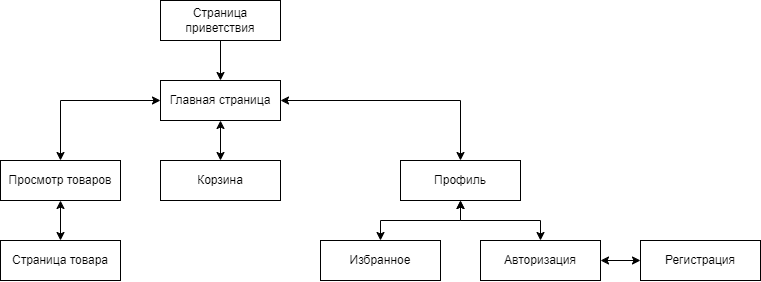
****

Рисунок 2.2 – Схема навигации для гостя

Страница приветствия показывается всем пользователям при первом входе.

На рисунке 2.3 представлена схема навигации пользователя.

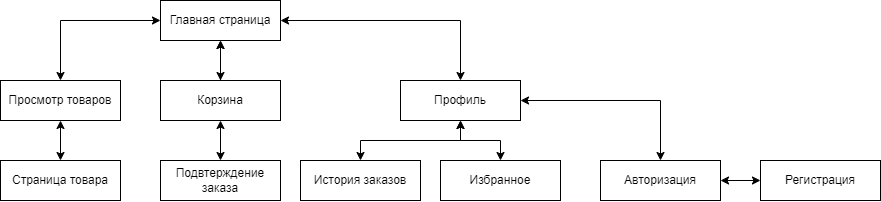


Рисунок 2.3 – Схема навигации для пользователя

На рисунке 2.4 представлена схема навигации администратора.

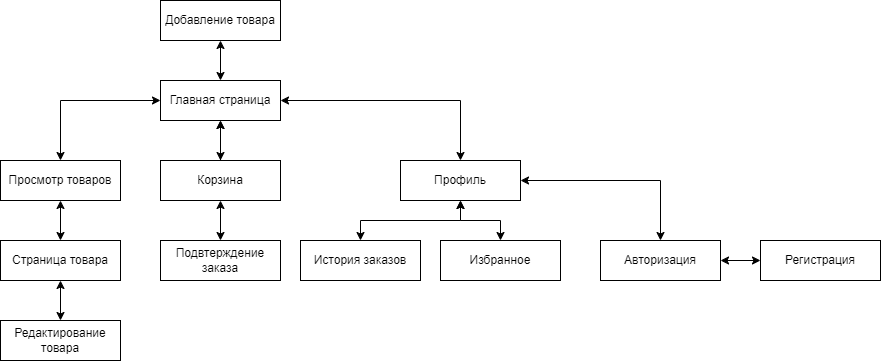


Рисунок 2.4 – Схема навигации администратора

Диаграмма вариантов использований представлена в приложении А.

# 3 Реализация приложения

Реализация приложения - это процесс преобразования проекта программного обеспечения в работающий продукт. Это включает в себя написание кода, интеграцию различных компонентов системы, тестирование для обнаружения и исправления ошибок, а также оптимизацию для улучшения производительности. Реализация приложения требует глубоких технических знаний и навыков в области программирования, а также понимания бизнес-требований и пользовательских потребностей. Это ключевой этап в жизненном цикле разработки программного обеспечения, который влияет на качество и успешность конечного продукта.

## 3.1 Технические средства разработки

Для реализации курсового проекта будет использован язык программирования Dart вместе с фремворком Flutter.

Dart - это язык программирования, разработанный компанией Google. Он был представлен в 2011 году и позиционируется как замена или альтернатива JavaScript. Dart - это объектно-ориентированный, веб-фреймворк, сценарный, императивный, рефлексивный язык программирования, язык функционального программирования и мультипарадигмальный.

Основной сферой применения Dart на сегодняшний день является разработка графических приложений с помощью кроссплатформенного фреймворка Flutter. Dart предлагает современные функции, такие как безопасность от null и сопоставление с образцом. Dart также поддерживает горячую перезагрузку, что позволяет видеть результаты изменений в коде немедленно в работающем приложении.

Flutter - это открытый UI-инструментарий, разработанный Google для создания красивых, компилируемых нативно приложений для мобильных устройств, веб и настольных компьютеров из одной кодовой базы. Flutter работает с существующим кодом и используется разработчиками и организациями по всему миру.

Основные преимущества Flutter включают быструю разработку благодаря возможности горячей перезагрузки, которая позволяет видеть результаты изменений в коде практически мгновенно, не теряя состояния. Кроме того, Flutter позволяет контролировать каждый пиксель для создания настраиваемых, адаптивных дизайнов, которые выглядят и чувствуются отлично на любом экране.

Flutter также поддерживает развертывание на множестве устройств из одной кодовой базы, включая мобильные, веб, настольные и встроенные устройства3. Это делает его мощным инструментом для разработки кросс-платформенных приложений. Flutter поддерживается и используется Google, доверяют ему известные бренды по всему миру и поддерживается сообществом разработчиков из разных стран.

База данных и механизмы аутентификации были разработаны c помощью Firebase. Локальная бд разработана с помощью Hive.

Firebase - это платформа разработки приложений от Google, которая помогает разработчикам создавать, улучшать и масштабировать приложения. Firebase предлагает ряд сервисов, включая Authentication, Firestore и Cloud Storage.

Firebase Authentication предоставляет бэкенд, SDK и готовые UI-библиотеки для аутентификации пользователей в вашем приложении. Он поддерживает аутентификацию с использованием паролей, номеров телефонов, популярных федеративных поставщиков идентификации, таких как Google, Facebook и Twitter.

Firestore - это гибкая, масштабируемая база данных для мобильных, веб- и серверных разработок от Firebase и Google Cloud. Firestore сохраняет ваши данные в синхронизации на всех клиентских приложениях с помощью слушателей в реальном времени и предлагает поддержку в режиме офлайн для мобильных и веб-приложений.

Cloud Storage for Firebase - это мощный, простой и экономичный сервис хранения объектов, построенный для работы в масштабе Google. SDK Firebase для Cloud Storage добавляют безопасность Google к загрузке и скачиванию файлов для ваших Firebase-приложений, независимо от качества сети. Вы можете использовать наши клиентские SDK для хранения изображений, аудио, видео или другого контента, созданного пользователями.

Для данного приложения используются Firebase Authentication, Firebase Cloud и Firebase Storage.

Hive - это легковесная и быстрая база данных типа ключ-значение, написанная на чистом Dart. Она вдохновлена Bitcask и предназначена для использования с Flutter.

Hive предлагает ряд преимуществ, включая отличную производительность и простой, мощный и интуитивно понятный API. Она поддерживает кросс-платформенность (мобильные устройства, настольные компьютеры, браузеры) и не имеет нативных зависимостей. Hive также включает в себя сильное шифрование.

Одной из особенностей Hive является то, что она позволяет использовать ее как обычную карту, и не требуется ожидать завершения Future. Кроме того, Hive поддерживает хранение не только примитивов, списков и карт, но и любых объектов Dart.

## 3.2 Разработка мобильного приложения

Структура приложения состоит из нескольки папок: Models, ViewModels, Views. Views содержит страницы и bloc-элементы приложения. ViewModels содержит основную бизнес-логику приложения, а models содержит модели и методы доступа к ним.

При реализации приложения использовано асинхронное программирование.

Данные хранятся локально и в удаленном хранилище Firestore. Для увеличение быстродействия данные в локальной БД синхронизируются с данными удаленной БД, после чего в приложении используется локальная БД.

Ниже приведена реализация метода, который синхронезирует локальную и удаленную БД. Метод вызывается после входа пользователя в аккаунт. Метод проверяет, что пользователь авторизован, открывает локальную бд, и добавляет все записи пользователя из удаленной БД в нее.

Future<void> refreshOrders() async {

User? user = FirebaseAuth.instance.currentUser;

if (user != null) {

Box<OrderEntity> box;

if (Hive.isBoxOpen(orderBoxName)) {

box = Hive.box<OrderEntity>(orderBoxName);

} else {

box = await Hive.openBox<OrderEntity>(orderBoxName);

}

await box.clear();

CollectionReference userOrders = FirebaseFirestore.instance.collection('users').doc(user.uid).collection('orders');

QuerySnapshot snapshot = await userOrders.get();

for (var doc in snapshot.docs) {

OrderEntity orderEntity = OrderEntity.fromMap(doc.data() as Map<String, dynamic>);

await box.add(orderEntity);

}

}

}

Листинг 3.1 – Синхронизация БД

Когда пользователь добавляет заказ, система сохраняет информацию о заказе в двух местах: в локальной базе данных на устройстве пользователя и в удаленной базе данных, которая обслуживается на сервере.

В момент выхода пользователя из системы, локальная база данных очищается. Это означает, что все данные, которые были сохранены локально на устройстве пользователя, удаляются. Это делается для обеспечения безопасности и конфиденциальности данных пользователя. Кроме того, это помогает убедиться, что в локальной базе данных не сохраняется устаревшая или невалидная информация.

Когда пользователь входит в систему снова, ему отображаются только его заказы. Это достигается путем запроса данных о заказах пользователя из удаленной базы данных и загрузки этих данных в локальную базу данных на устройстве пользователя. Таким образом, пользователь всегда видит актуальную информацию о своих заказах, независимо от того, с какого устройства он входит в систему.

Листинг всего класса для работы с заказами представлен в приложении Б.

# 4 Тестирование приложения

Тестирование мобильного программного обеспечения - это процесс проверки функциональности, производительности, безопасности и других ключевых аспектов мобильных приложений перед их выпуском. Это включает в себя проверку приложения на различных устройствах, операционных системах и конфигурациях, чтобы убедиться, что оно работает корректно во всех возможных сценариях. Тестирование может включать в себя автоматические и ручные методы, а также различные типы тестов, такие как функциональное тестирование, тестирование производительности, тестирование безопасности и тестирование пользовательского интерфейса. Цель тестирования мобильного ПО - обеспечить высокое качество приложения и предотвратить проблемы, которые могут негативно повлиять на пользовательский опыт.

Ниже приведен результат тстирования страницы авторизации, рисунок 4.1. При вводе некорректных данных, появляется попап, сообщаеющий об этом.

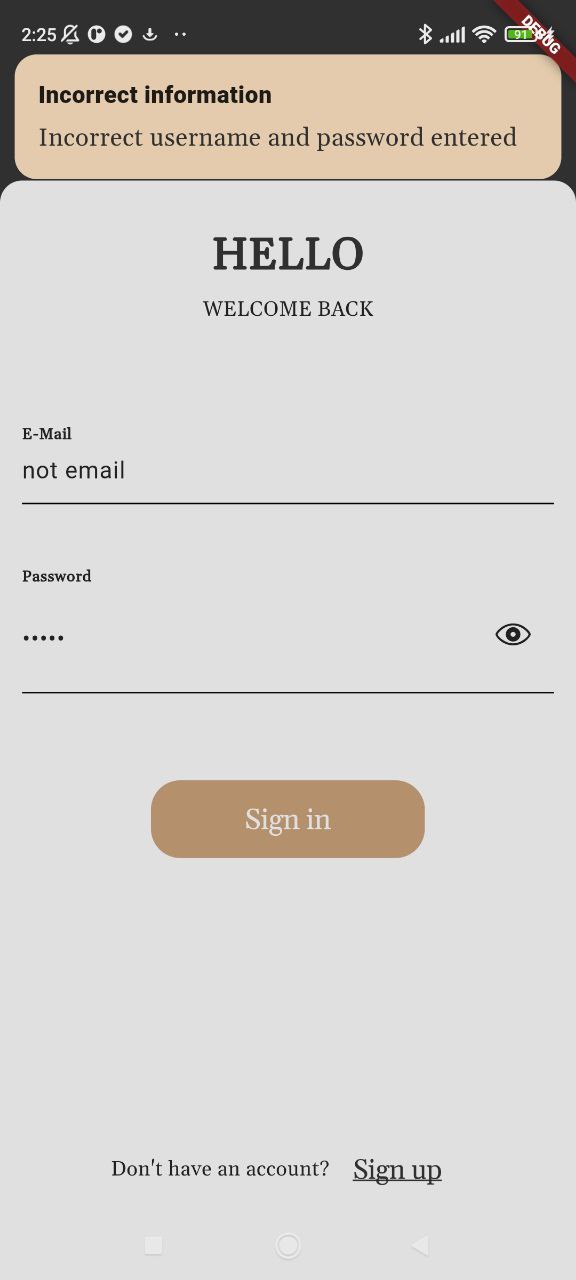


Рисунок 4.1 – Тестирование авторизации

Далее приведен тест страницы регистрации. При попытке ввести некорректную информацию пользователь также получает ошибку, рисунок 4.2.

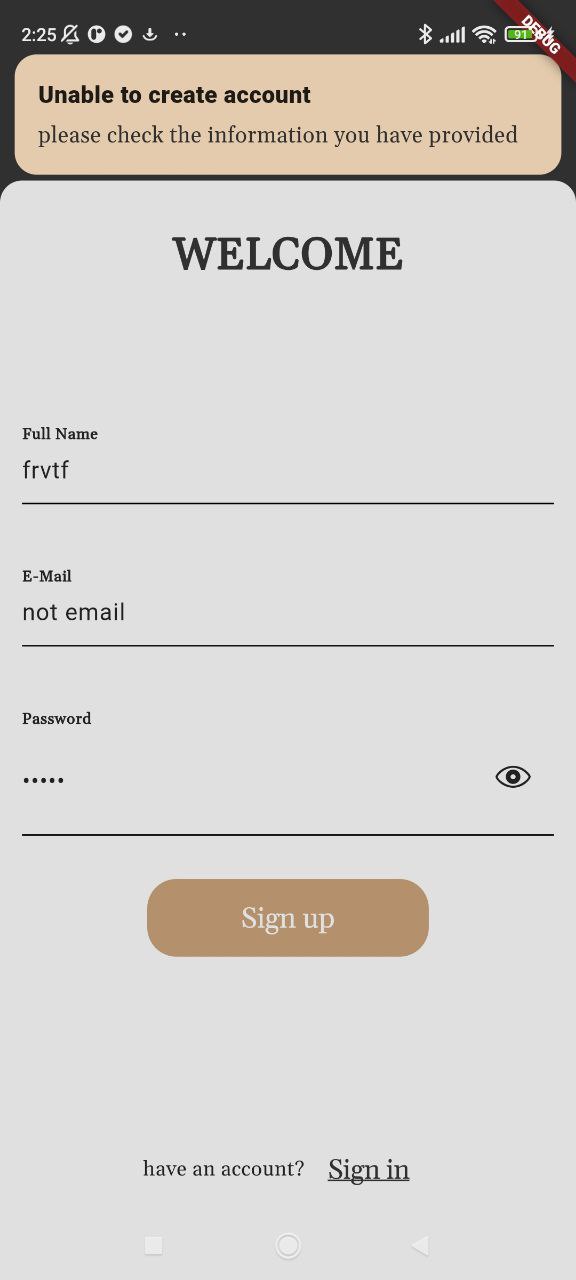


Рисунок 4.2 – Тестирование регистрации

# 5 Руководство по использованию

# В данном разделе будет описано руководство по использованию для гостя, пользователя и администратора.

## 5.1 Руководство гостя

После экрана приветствия пользователь попадает на главную страницу, рисунок 5.1.

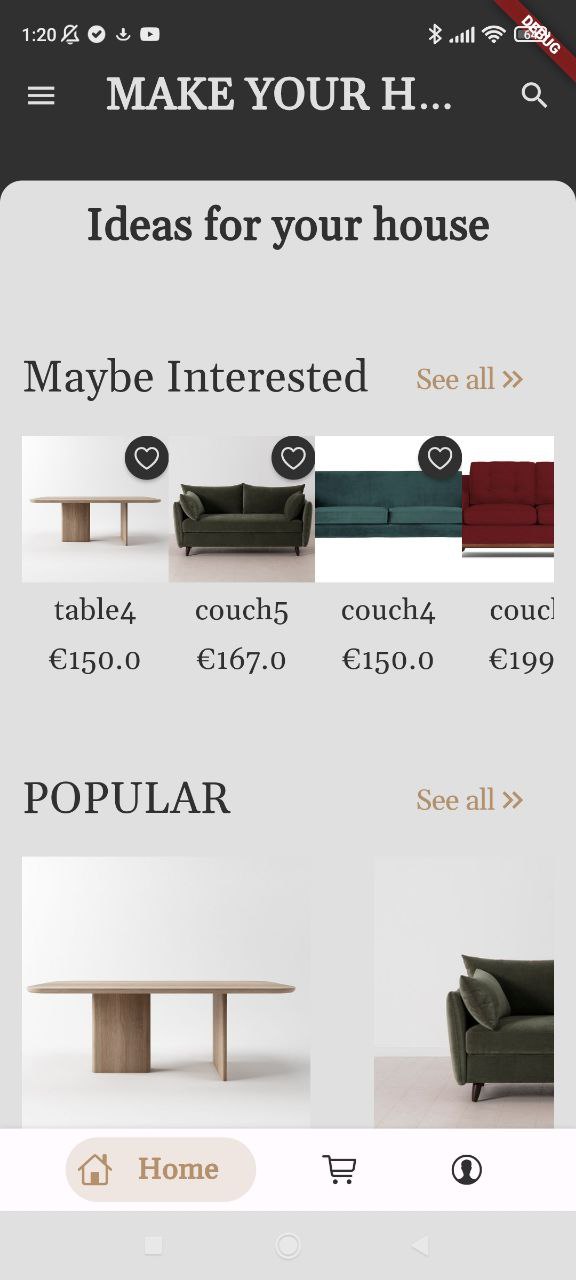


Рисунок 5.1 – Главная страница

Тут пользователь может перейти на страницу каталога, рисунок 5.2.

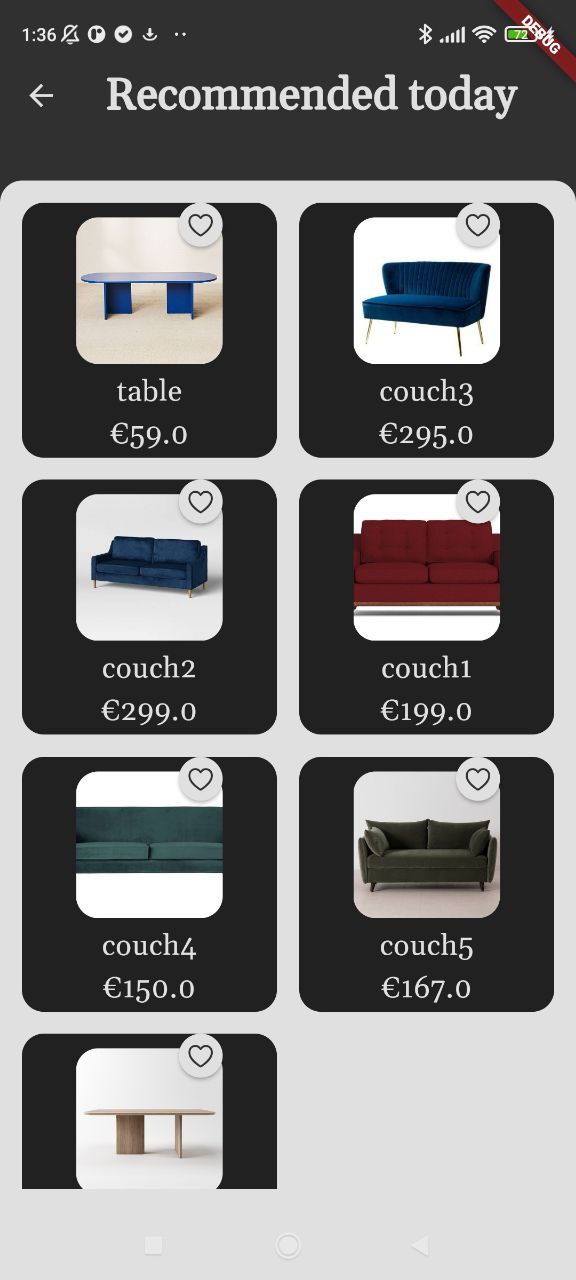


Рисунок 5.2 – Страница каталога

Пользователь может также перейти к поиску, рисунок 5.3.

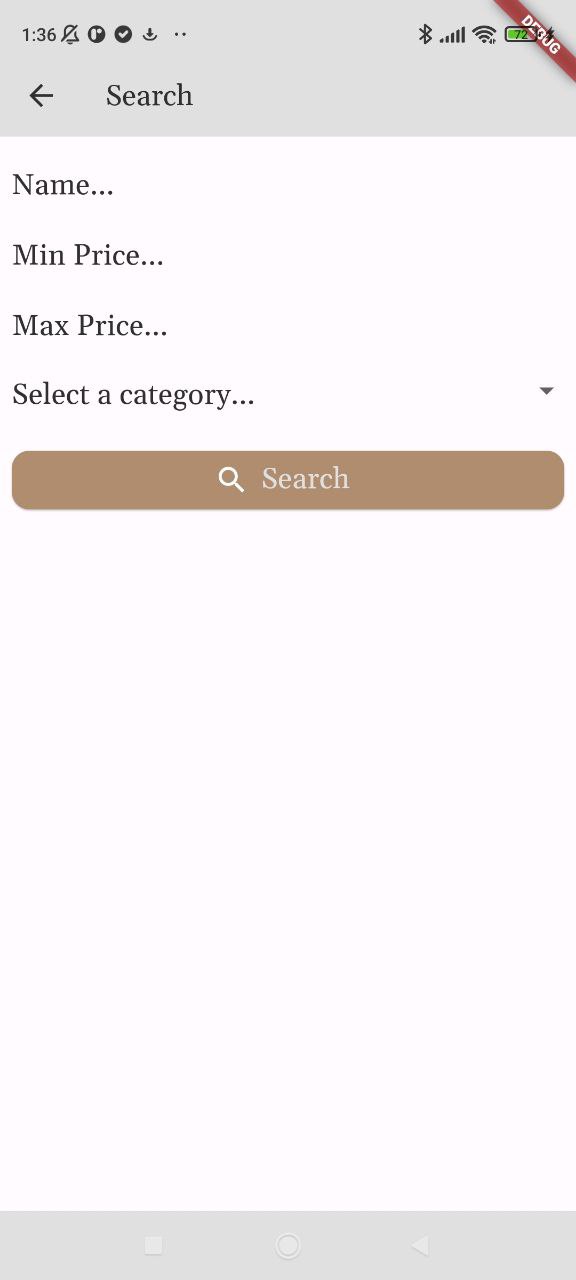


Рисунок 5.3 – Страница поиска

Нажав на товар пользователь может посмотреть информацию о нем открыв страницу товара, где пользователь может добавить его в корзину или в избранное, рисунок 5.4.



Рисунок 5.4 – Страница товара

Из главной страницы можно перейти к странице профиля, где можно будет выбрать опцию входа. Страница авторизации представлена на рисунке 5.5.

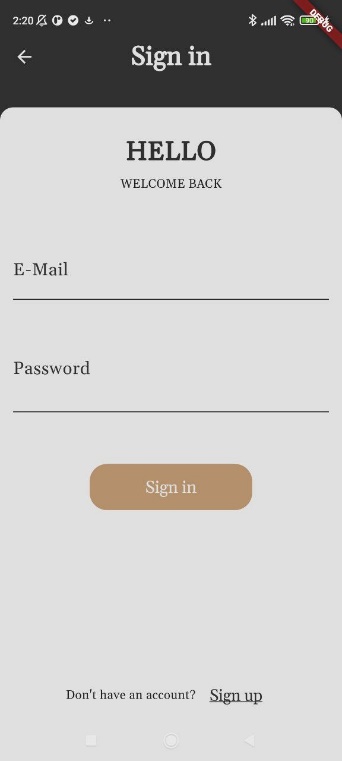


Рисунок 5.5 – Страница авторизации

Нажав на опцию снизу можно перейти к регистрации. Страница регистрации представлена на рисунке 5.6.

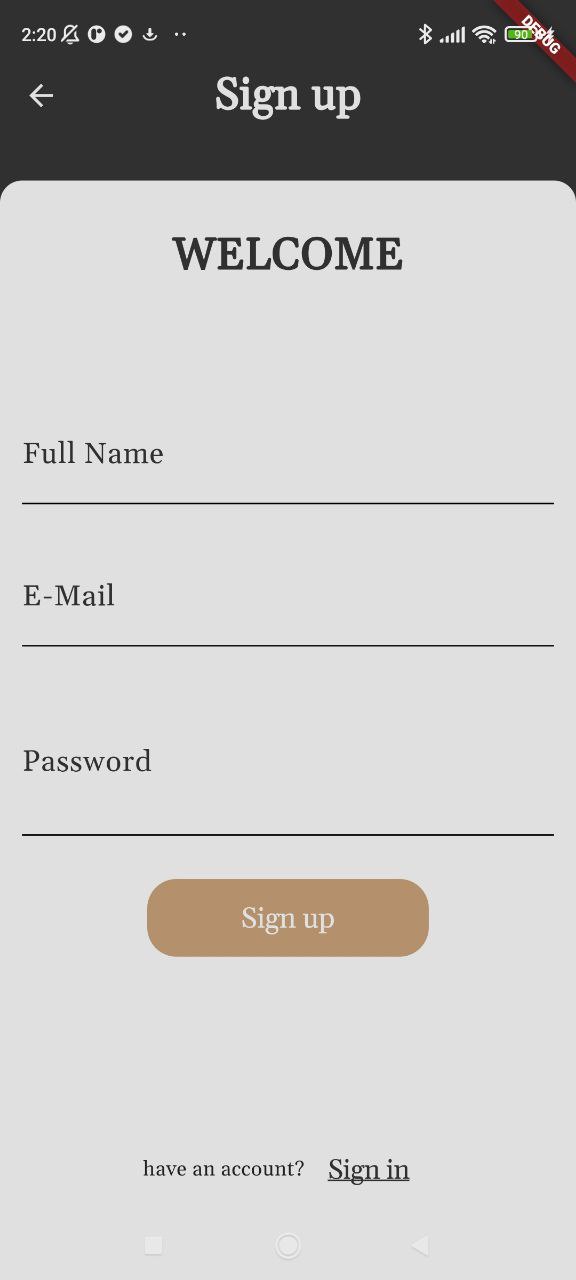


Рисунок 5.6 – Страница регистрации

После успешного входа пользователь приобретает роль пользователя или администратора.

## 5.2 Руководство пользователя

После входа пользователь может изменить свою фотографию, имя, а также посмотреть свои избранные товары и историю заказов, рисунок 5.7.

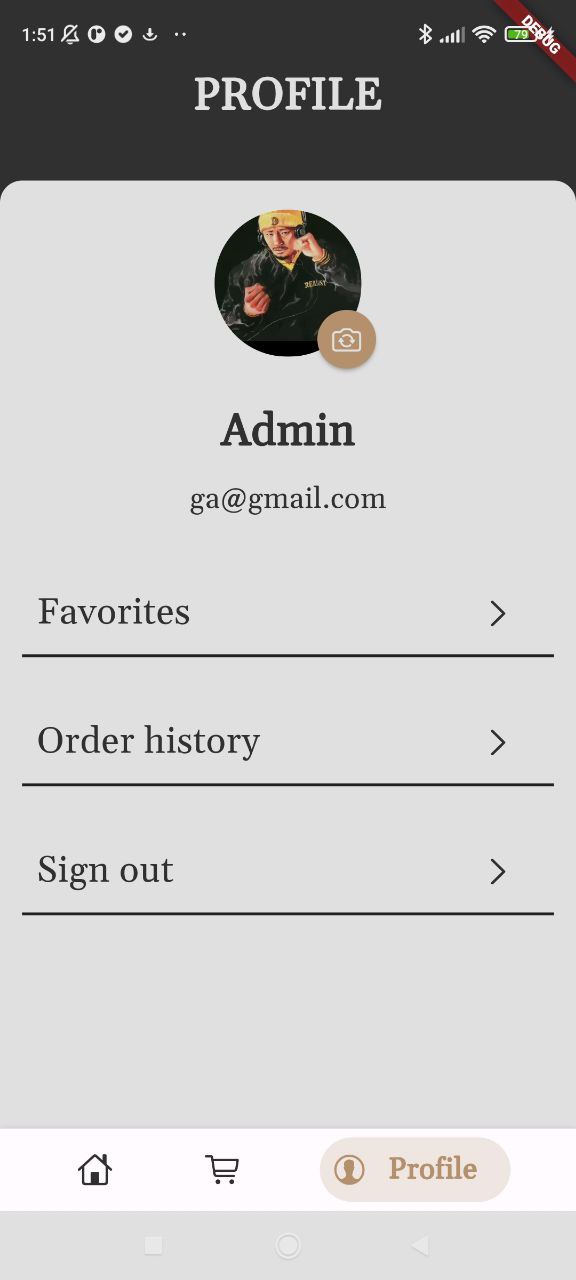


Рисунок 5.7 – Страница профиля

Используя таб-бар снизу, пользователь может перейти к странице своей корзины, рисунок 5.8.



Рисунок 5.8 – Страница корзины

После этого пользователь может перейти к странице подтверждения заказа, рисунок 5.9.

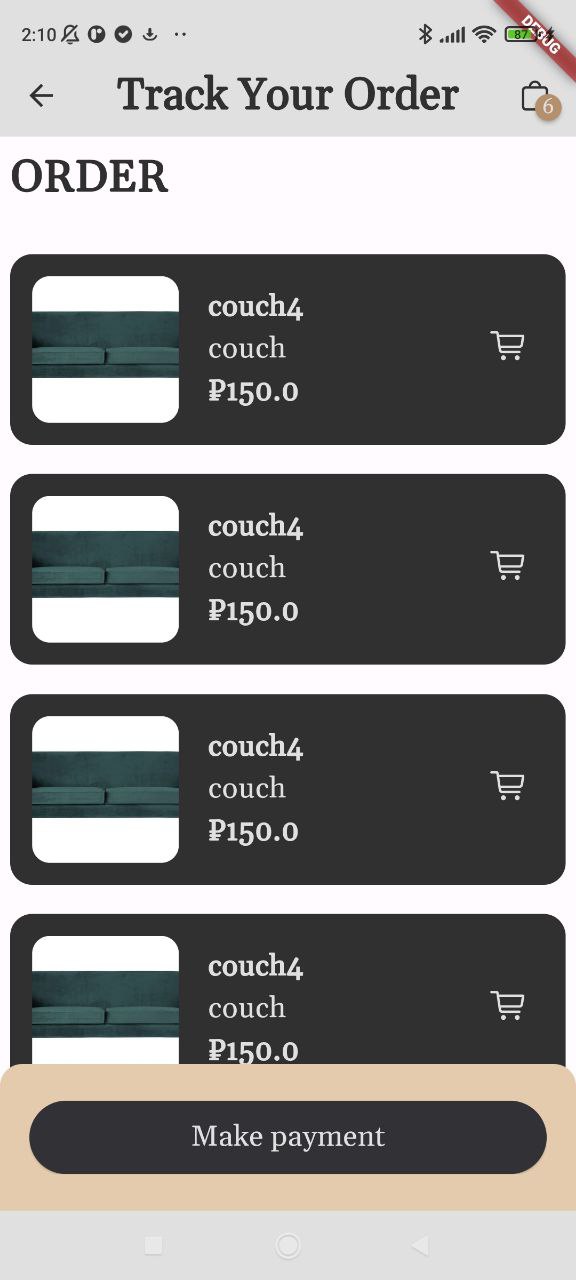


Рисунок 5.9 – Страница подтверждения заказа

После этого пользователю будет представлено подтверждение оплаты, рисунок 5.10.

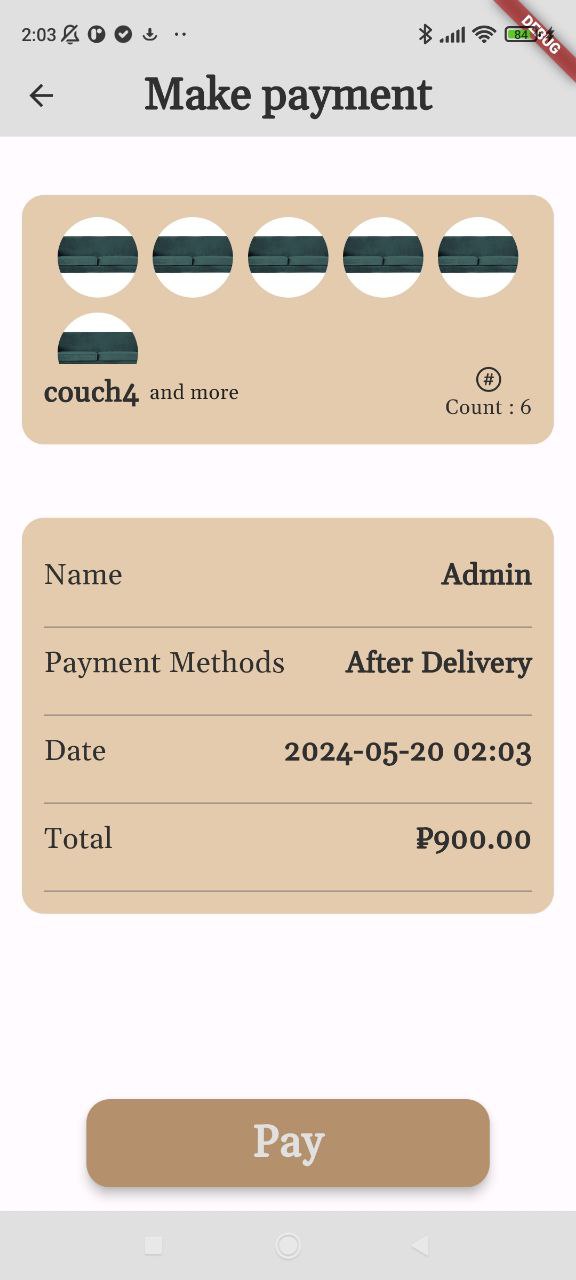


Рисунок 5.10 – Подтверждение оплаты

Нажав на кнопку снизу заказ будет зарегистрирован и появится в истории заказов.

## 5.3 Руководство администратора

Нажав на опции меню сверху слева администратор может перейти к странице добавления нового товара, рисунок 5.11.

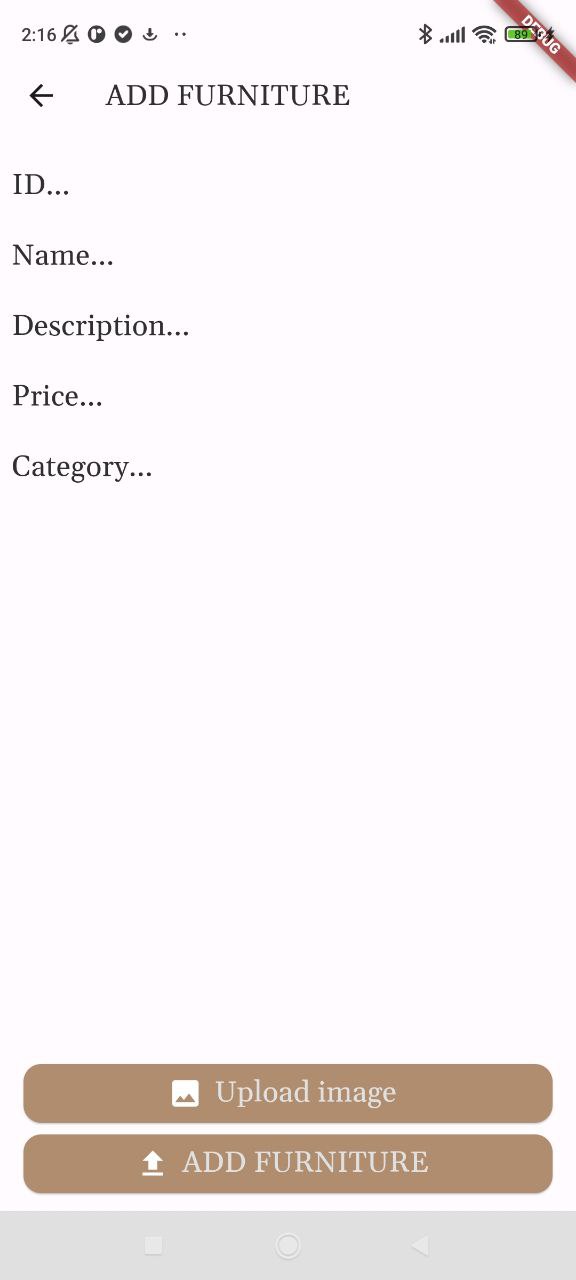


Рисунок 5.11 – Страница добавления товара

Администратор может также удалять и изменять товары используя соответствующие кнопки на странице товара.

# Заключение

В результате выполнения данной курсовой работы было разработано мобильное приложение для интернет магазина мебели.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* авторизация;
* просмотр и поиск товаров;
* добавление товаров в корзину, оформление заказа;
* добавление товаров в избанное;
* просмотр своих заказов;
* изменение профиля;

Администратору предоставляет следующие функциональные возможности:

* авторизация;
* просмотр и поиск товаров;
* добавление товаров в корзину, оформление заказа;
* добавление товаров в избанное;
* просмотр своих заказов;
* изменение профиля;
* добавление, удаление и изменение товаров;

Гость имеет возможность:

* регистрация;
* авторизация;
* просмотр и поиск товаров;
* добавление товаров в избанное;

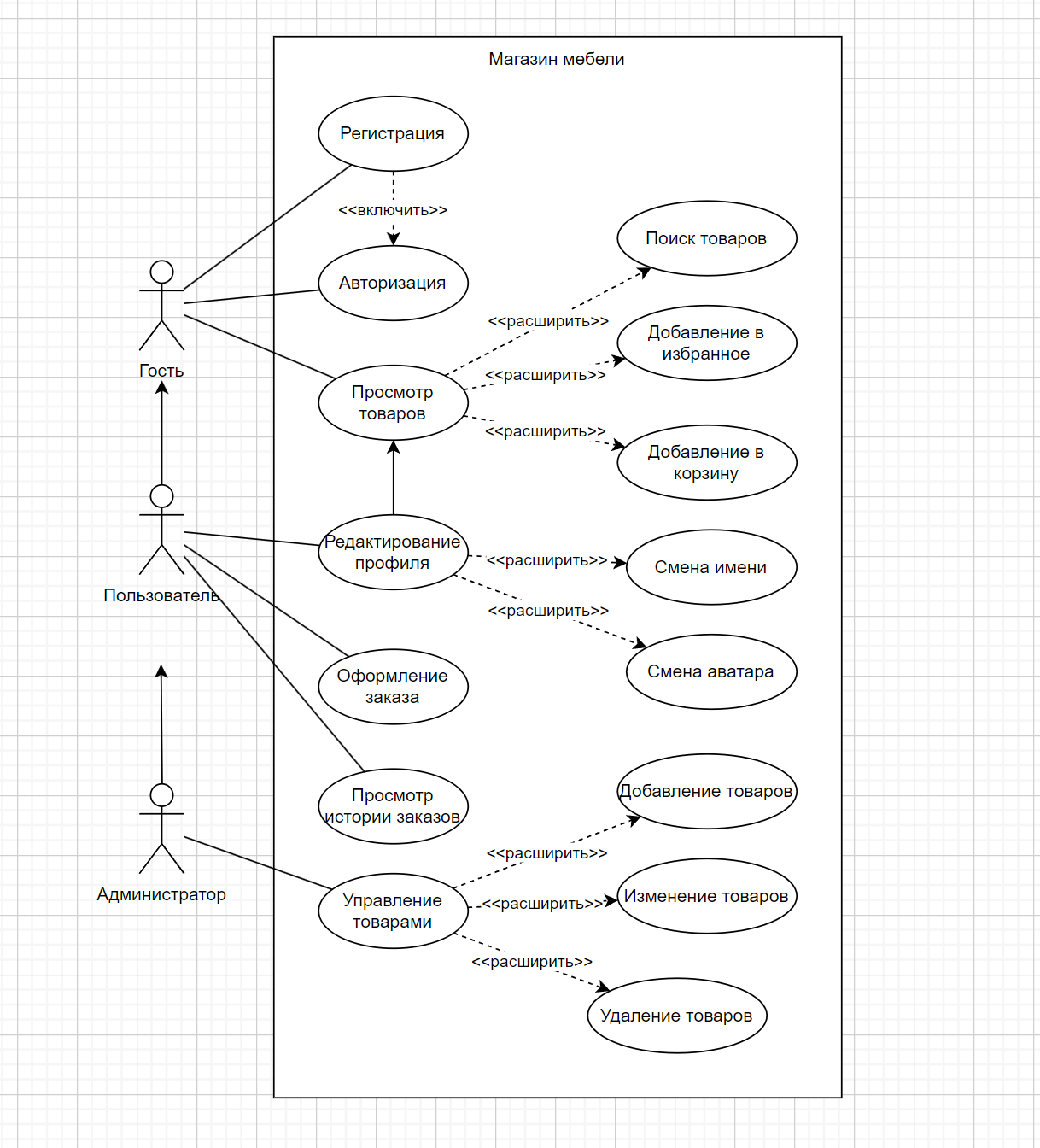
Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список использованных источников

1. Flutter Documentation [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/ – Дата доступа: 14.03.2024
2. Dart Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://dart.dev/guides/ – Дата доступа: 22.03.2024;
3. Firebase Documentation [Электронный ресурс]. – Режим доступа: https://firebase.google.com/docs/ – Дата доступа: 20.04.2024;
4. Hive Documentation [Электронный ресурс]. – Режим доступа: https://docs.hivedb.dev/#/ – Дата доступа: 26.04.2024.
5. SQLite Documentation [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/docs.html – Дата доступа: 10.05.2024.

# ПРИЛОЖЕНИЕ А - UML диаграмма использований



# ПРИЛОЖЕНИЕ Б - Класс PaymentScreen

import 'package:cached\_network\_image/cached\_network\_image.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_application\_ecommerce/Model/GetX/Controller/duplicate\_controller.dart';

import 'package:flutter\_application\_ecommerce/Model/GetX/Controller/profile\_controller.dart';

import 'package:flutter\_application\_ecommerce/Model/Tools/Color/color.dart';

import 'package:flutter\_application\_ecommerce/Model/Tools/Entities/OrderEntity/order\_entity.dart';

import 'package:flutter\_application\_ecommerce/Model/Tools/JsonParse/product\_parse.dart';

import 'package:flutter\_application\_ecommerce/Model/Widget/widget.dart';

import 'package:flutter\_application\_ecommerce/View/RootScreen/root.dart';

import 'package:flutter\_svg/flutter\_svg.dart';

import 'package:get/get.dart';

class PaymentScreen extends StatelessWidget {

const PaymentScreen(

{super.key,

required this.totalPrice,

required this.productList,

required this.addressDetail});

final String totalPrice;

final List<ProductEntity> productList;

final String addressDetail;

@override

Widget build(BuildContext context) {

final duplicateController = Get.find<DuplicateController>();

final profileController = Get.find<ProfileController>();

final colors = duplicateController.colors;

final textStyle = duplicateController.textStyle;

final paymentFunctions = duplicateController.paymentFunctions;

final DateTime dateTime = DateTime.now();

return Scaffold(

floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,

floatingActionButton: SizedBox(

height: 60,

width: Get.mediaQuery.size.width \* 0.7,

child: FloatingActionButton.extended(

backgroundColor: colors.primary,

onPressed: () {

showCupertinoDialog(

context: context,

builder: (context) {

return CupertinoAlertDialog(

title: Text(

"Pay",

style: textStyle.bodyNormal,

),

content: Text(

"Do you want pay?",

style: textStyle.bodyNormal,

),

actions: [

CupertinoButton(

child: const Text("No"),

onPressed: () {

Get.back();

},

),

CupertinoButton(

child: const Text("Yes"),

onPressed: () async {

await profileController.orderFunctions.addToOrderBox(

orderEntity: OrderEntity(

time: dateTime,

totalPrice: totalPrice,

productList: productList));

bool isCommpleated = await duplicateController

.cartFunctions

.clearCartBox();

if (isCommpleated) {

Get.offAll(const RootScreen(

index: 1,

));

snackBar(

title: "Pay",

message: "Successfully payed",

textStyle: textStyle,

colors: colors);

}

},

),

],

);

},

);

},

label: Text(

"Pay",

style: textStyle.titleLarge.copyWith(color: colors.whiteColor),

)),

),

appBar: AppBar(

backgroundColor: colors.whiteColor,

foregroundColor: colors.blackColor,

centerTitle: true,

automaticallyImplyLeading: true,

title: Text(

"Make payment",

style: textStyle.titleLarge,

),

),

body: Padding(

padding: const EdgeInsets.all(15),

child: SingleChildScrollView(

physics: duplicateController.uiDuplicate.defaultScroll,

child: Column(

children: [

// Align(

// alignment: Alignment.center,

// child: SvgPicture.string(

// paymentFunctions.createBarcode(),

// ),

// ),

duplicateContainer(

colors: colors,

widget: Column(

children: [

SizedBox(

height: 100,

width: Get.mediaQuery.size.width \* 0.8,

child: GridView.builder(

physics: duplicateController.uiDuplicate.defaultScroll,

itemCount: productList.length,

gridDelegate:

const SliverGridDelegateWithFixedCrossAxisCount(

crossAxisCount: 5,

mainAxisSpacing: 10,

crossAxisSpacing: 10),

itemBuilder: (context, index) {

return SizedBox(

height: 40,

child: CircleAvatar(

backgroundColor: colors.whiteColor,

foregroundImage: CachedNetworkImageProvider(

productList[index].imageUrl),

),

);

},

),

),

const SizedBox(

width: 10,

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

SizedBox(

width: Get.mediaQuery.size.width \* 0.48,

child: Row(

children: [

Text(

productList[0].name.substring(

0,

productList[0].name.length > 8

? 8

: productList[0].name.length) +

(productList[0].name.length > 8

? "..."

: ""),

style: textStyle.bodyNormal

.copyWith(fontWeight: FontWeight.bold),

overflow: TextOverflow.clip,

textAlign: TextAlign.start,

),

const SizedBox(

width: 7,

),

Text(

"and more",

style: textStyle.bodySmall,

),

],

)),

Column(

children: [

Icon(

CupertinoIcons.number\_circle,

color: colors.blackColor,

size: 20,

),

Text(

"Count : ${productList.length}",

style: textStyle.bodySmall

.copyWith(color: colors.blackColor),

),

],

)

],

)

],

),

),

duplicateContainer(

colors: colors,

widget: Column(

children: [

duplicateRowItem(

colors: colors,

prefix: Text(

"Name",

style: textStyle.bodyNormal,

),

suffix: Text(

profileController.information.name,

style: textStyle.bodyNormal

.copyWith(fontWeight: FontWeight.bold),

)),

duplicateRowItem(

colors: colors,

prefix: Text(

"Payment Methods",

style: textStyle.bodyNormal,

),

suffix: Text(

"After Delivery",

style: textStyle.bodyNormal

.copyWith(fontWeight: FontWeight.bold),

)),

duplicateRowItem(

colors: colors,

prefix: Text(

"Date",

style: textStyle.bodyNormal,

),

suffix: Text(

dateTime.toString().substring(0, 16),

style: textStyle.bodyNormal

.copyWith(fontWeight: FontWeight.bold),

)),

duplicateRowItem(

colors: colors,

prefix: Text(

"Total",

style: textStyle.bodyNormal,

),

suffix: Text(

"₽$totalPrice",

style: textStyle.bodyNormal

.copyWith(fontWeight: FontWeight.bold),

)),

],

)),

],

),

),

),

);

}

Widget duplicateContainer(

{required Widget widget, required CustomColors colors}) {

return Container(

width: Get.mediaQuery.size.width,

margin: const EdgeInsets.only(top: 25, bottom: 25),

padding: const EdgeInsets.all(15),

decoration: BoxDecoration(

color: colors.gray,

borderRadius: BorderRadius.circular(15),

),

child: widget);

}

Widget duplicateRowItem(

{required Widget prefix,

required Widget suffix,

required CustomColors colors}) {

return Column(

children: [

Padding(

padding: const EdgeInsets.only(top: 10, bottom: 10),

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [prefix, suffix],

),

),

const SizedBox(

height: 10,

),

Divider(

thickness: 0.5,

height: 1,

color: colors.captionColor,

)

],

);

}